1.Do implementation of functions to calculate mean, median, mode, variance, and standard deviation.

```
import statistics

# Function to calculate the mean
def calculate_mean(data):
    return sum(data) / len(data)

# Function to calculate the median
def calculate_median(data):
    return statistics.median(data)

# Function to calculate the mode
def calculate_mode(data):
    return statistics.mode(data)

# Function to calculate the variance
def calculate_variance(data):
    return statistics.variance(data)

# Function to calculate the standard deviation
def calculate_standard_deviation(data):
    return statistics.stdev(data)

# Example usage
data = [10, 20, 20, 30, 40, 50]

mean = calculate_mean(data)
median = calculate median(data)
mode = calculate_mode(data)
variance = calculate_variance(data)
std_deviation = calculate_standard_deviation(data)

print("Mean:", mean)
print("Median:", median)
print("Mode:", mode)
print("Variance:", variance)
print("Standard Deviation:", std_deviation)
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python mod.py
Mean: 28.333333333333332
Median: 25.0
Mode: 20
Variance: 216.66666666666666
Standard Deviation: 14.719601443879744
```

Demonstrate the use of log(), log10(), and log2() functions on a set of positive numbers.

```
import math

# Set of positive numbers
```

```
numbers = [1, 2, 5, 10, 20, 100]

# Using log(), log10(), and log2()
log_values = {
    "log(x) (Natural Log)": [math.log(x) for x in numbers],
    "log10(x)": [math.log10(x) for x in numbers],
    "log2(x)": [math.log2(x) for x in numbers],
}

# Display results
for label, values in log_values.items():
    print(f"{label}: {values}")
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python log.py
log(x) (Natural Log): [0.0, 0.6931471805599453, 1.6094379124341003, 2.302585092994046, 2.995732273553991, 4.605170185988092]
log10(x): [0.0, 0.3010299956639812, 0.6989700043360189, 1.0, 1.3010299956639813, 2.0]
log2(x): [0.0, 1.0, 2.321928094887362, 3.321928094887362, 4.321928094887363, 6.643856189774724]
```

Write a program to compute the following using built-in math functions: (i Square root of a number (ii Exponential value of a number (iii Power of a number (x^y).

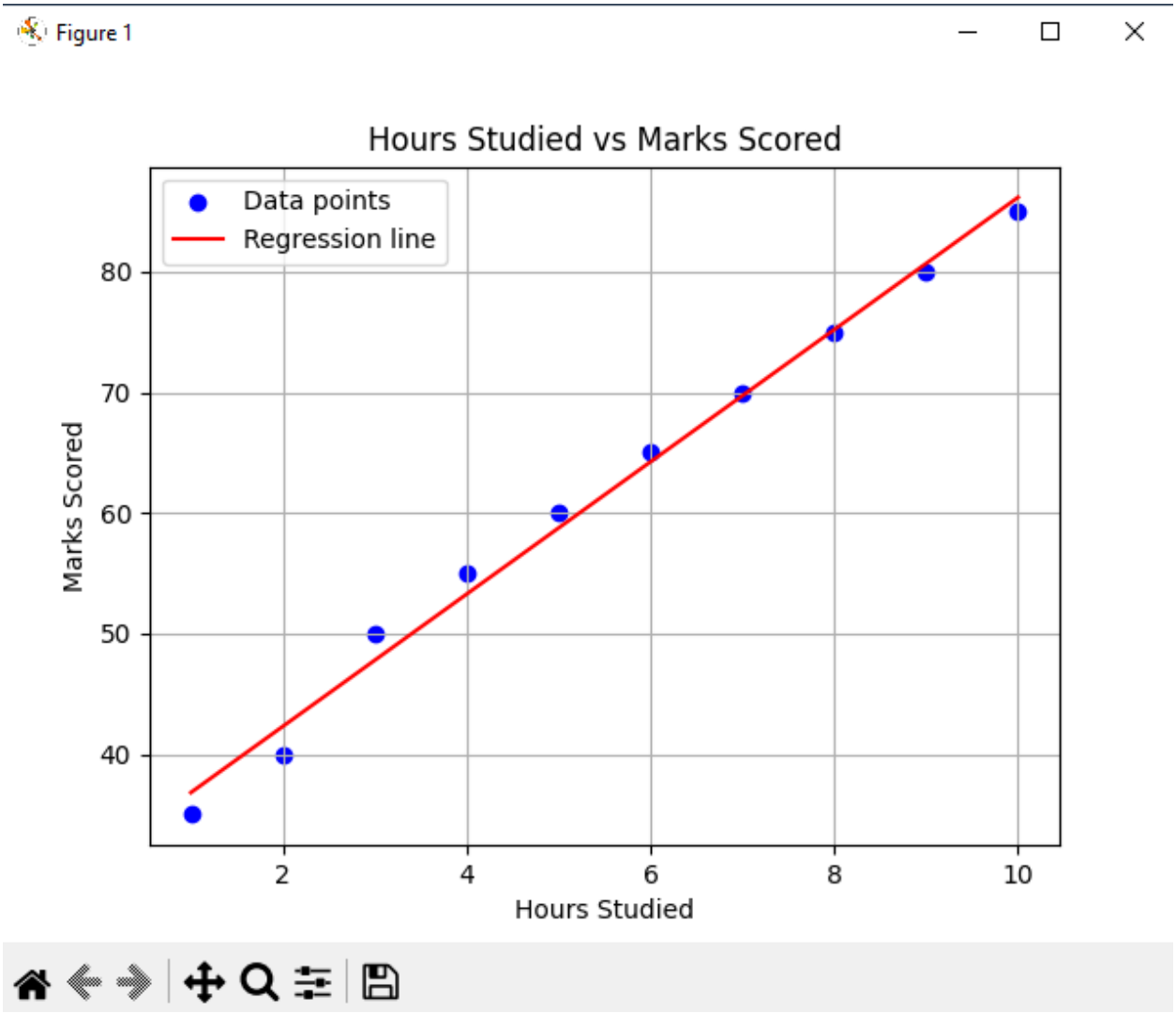
```
import math

# Function to calculate square root
def calculate_square_root(number):
    return math.sqrt(number)

# Function to calculate exponential value (e^x)
def calculate_exponential(number):
    return math.exp(number)

# Function to calculate power of a number (x^y)
def calculate_power(base, exponent):
    return math.pow(base, exponent)

# Example usage
num = 16  # Number for square root and exponential example
base = 2  # Base for power calculation
exponent = 3  # Exponent for power calculation

# Calculating square root
sqrt_result = calculate_square_root(num)

# Calculating exponential
exp_result = calculate_exponential(num)

# Calculating power
pow result = calculate power(base, exponent)

# Display results
print(f"Square root of {num}: {sqrt_result}")
print(f"Exponential of {num} (e^{num}): {exp_result}")
print(f"{base} raised to the power of {exponent} ({base}^{exponent}):
{pow_result}")
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>built.py
Square root of 16: 4.0
Exponential of 16 (e^16): 8886110.520507872
2 raised to the power of 3 (2^3): 8.0
```

Demonstrate the use of correlation and covariance functions on two datasets representing "hours studied" and "marks scored". Also make a plot a scatter plot with a regression line to visually show the correlation between *hours studied* and *marks scored.*

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Sample data
hours_studied = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
marks_scored = np.array([35, 40, 50, 55, 60, 65, 70, 75, 80, 85])

# Calculate covariance matrix
cov_matrix = np.cov(hours_studied, marks_scored)
covariance = cov_matrix[0, 1]  # Covariance between hours_studied and
marks_scored

# Calculate Pearson correlation coefficient
correlation, p_value = stats.pearsonr(hours_studied, marks_scored)

print(f"Covariance: {covariance}")
print(f"Correlation coefficient: {correlation}")

# Scatter plot
plt.scatter(hours_studied, marks_scored, color='blue', label='Data points')

# Regression line
slope, intercept, r_value, p_value, std_err = stats.linregress(hours_studied,
marks_scored)
regression_line = slope * hours_studied + intercept
plt.plot(hours_studied, regression_line, color='red', label='Regression
line')

plt.title('Hours Studied vs Marks Scored')
plt.xlabel('Hours Studied')
plt.ylabel('Marks Scored')
plt.legend()
plt.grid(True)
plt.show()
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python correlation.py
Covariance: 50.27777777777777
Correlation coefficient: 0.99587439466412
```



Given student marks of a class, calculate the minimum, maximum, sum, and average using built-in statistical functions.

```
# Sample list of student marks
marks = [78, 85, 62, 90, 55, 89, 73, 94, 88, 76]

# Calculate minimum marks
min_marks = min(marks)

# Calculate maximum marks
max_marks = max(marks)
```

```
# Calculate sum of marks
total_marks = sum(marks)

# Calculate average marks
average_marks = total_marks / len(marks)

# Display results
print(f"Minimum marks: {min_marks}")
print(f"Maximum marks: {max_marks}")
print(f"Sum of marks: {total_marks}")
print(f"Average marks: {average_marks:.2f}")
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python statistical.py
Minimum marks: 55
Maximum marks: 94
Sum of marks: 790
Average marks: 79.00
```

Write a program to perform the following operations on a given string: (i Find length of the string (ii Convert to uppercase and lowercase (iii Count the number of occurrences of a given character

```
# Function to perform string operations
def string_operations(s, char_to_count):
    # (i) Find length of the string
    length = len(s)

    # (ii) Convert to uppercase and lowercase
    upper = s.upper()
    lower = s.lower()

    # (iii) Count the number of occurrences of a given character
    count_char = s.count(char_to_count)

    # Display results
    print(f"Original string: {s}")
    print(f"Length of the string: {length}")
    print(f"Uppercase: {upper}")
    print(f"Lowercase: {lower}")
    print(f"Occurrences of '{char_to_count}': {count_char}")


# Example usage
input_string = "Hello World! Welcome to Python Programming."
character = 'o'

string_operations(input_string, character)
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python upp.py
Original string: Hello World! Welcome to Python Programming.
Length of the string: 43
Uppercase: HELLO WORLD! WELCOME TO PYTHON PROGRAMMING.
Lowercase: hello world! welcome to python programming.
Occurrences of 'o': 6
```

Accept a string from the user and: (i Remove leading and trailing spaces (ii Replace all spaces with underscores _

```python
# Accept string input from the user
user_input = input("Enter a string: ")

# (i) Remove leading and trailing spaces
trimmed_string = user_input.strip()

# (ii) Replace all spaces with underscores
modified_string = trimmed_string.replace(' ', '_')

# Display the results
print("Original string:", repr(user_input))
print("After removing leading and trailing spaces:", repr(trimmed_string))
print("After replacing spaces with underscores:", repr(modified_string))
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python string.py
Enter a string: salu manoj
Original string: 'salu manoj'
After removing leading and trailing spaces: 'salu manoj'
After replacing spaces with underscores: 'salu_manoj'
```

Write a Python program using NumPy to create a 1D array of 10 random integers between 1 and 100.

- **Find the maximum, minimum, mean, and standard deviation.**

```python
import numpy as np

# Create a 1D array of 10 random integers between 1 and 100 (inclusive)
random_array = np.random.randint(1, 101, size=10)

# Calculate maximum
max_value = np.max(random_array)

# Calculate minimum
```

```
min_value = np.min(random_array)

# Calculate mean
mean_value = np.mean(random_array)

# Calculate standard deviation
std_deviation = np.std(random_array)

# Display results
print("Random Array:", random_array)
print(f"Maximum: {max_value}")
print(f"Minimum: {min_value}")
print(f"Mean: {mean_value:.2f}")
print(f"Standard Deviation: {std_deviation:.2f}")
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python 1D.py
Random Array: [23 52 97 85 55  5 42  1  7 80]
Maximum: 97
Minimum: 1
Mean: 44.70
Standard Deviation: 33.39
```

Implement vectorized operations on two arrays:

- **Element-wise addition, subtraction, multiplication, and division.**

```
import numpy as np

# Define two arrays
arr1 = np.array([10, 20, 30, 40, 50])
arr2 = np.array([2, 4, 6, 8, 10])

# Element-wise addition
addition = arr1 + arr2

# Element-wise subtraction
subtraction = arr1 - arr2

# Element-wise multiplication
multiplication = arr1 * arr2

# Element-wise division
division = arr1 / arr2

# Display results
print("Array 1:", arr1)
print("Array 2:", arr2)
print("Addition:", addition)
```

```
print("Subtraction:", subtraction)
print("Multiplication:", multiplication)
print("Division:", division)
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python 2D.py
Array 1: [10 20 30 40 50]
Array 2: [ 2  4  6  8 10]
Addition: [12 24 36 48 60]
Subtraction: [ 8 16 24 32 40]
Multiplication: [ 20  80 180 320 500]
Division: [5. 5. 5. 5. 5.]
```

Represent a dataset of 5 students with features [RollNo, Marks1, Marks2, Marks3] as a 2D array.

- **Compute the total and average marks for each student.**

```
2. import numpy as np

   # Dataset: [RollNo, Marks1, Marks2, Marks3]
   data = np.array([
       [1, 85, 78, 92],
       [2, 76, 88, 85],
       [3, 90, 92, 94],
       [4, 70, 75, 80],
       [5, 88, 85, 91]
   ])

   # Extract marks only (ignore RollNo column)
   marks = data[:, 1:]  # all rows, columns from index 1 onwards

   # Calculate total marks for each student
   total_marks = np.sum(marks, axis=1)

   # Calculate average marks for each student
   average_marks = np.mean(marks, axis=1)

   # Combine results with RollNo for display
   results = np.column_stack((data[:, 0], total_marks, average_marks))

   # Display the results
   print("RollNo  Total Marks  Average Marks")
   for row in results:
       print(f"{int(row[0]):<7} {int(row[1]):<11} {row[2]:.2f}")
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python 2D_array.py
RollNo  Total Marks  Average Marks
1       255         85.00
2       249         83.00
3       276         92.00
4       225         75.00
5       264         88.00
```

Given a 2D NumPy array representing sales of products in different regions, compute:

- **Total sales per product**
- **Total sales per region**

```
2. import numpy as np

   # Example 2D array: rows represent products, columns represent regions
   sales = np.array([
       [120, 150, 100],  # Product 1 sales in Region 1, 2, 3
       [80,  90,  60],   # Product 2 sales
       [200, 210, 180],  # Product 3 sales
       [150, 130, 170]   # Product 4 sales
   ])

   # Total sales per product (sum across columns)
   total_sales_per_product = np.sum(sales, axis=1)

   # Total sales per region (sum across rows)
   total_sales_per_region = np.sum(sales, axis=0)

   # Display results
   print("Total sales per product:")
   for i, total in enumerate(total_sales_per_product, start=1):
       print(f"Product {i}: {total}")

   print("\nTotal sales per region:")
   for j, total in enumerate(total_sales_per_region, start=1):
       print(f"Region {j}: {total}")
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python 2D_numpy.py
Total sales per product:
Product 1: 370
Product 2: 230
Product 3: 590
Product 4: 450

Total sales per region:
Region 1: 550
Region 2: 580
Region 3: 510
```

Create a 3D array of shape (2×3×4) using NumPy.

- **Display its shape, dimensions, and size.**

```python
import numpy as np

# Create a 3D array of shape (2, 3, 4) with random integers
array_3d = np.random.randint(1, 10, size=(2, 3, 4))

# Display the array (optional)
print("3D Array:\n", array_3d)

# Display shape
print("Shape:", array_3d.shape)

# Display number of dimensions
print("Number of dimensions (ndim):", array_3d.ndim)

# Display total size (number of elements)
print("Total size (number of elements):", array_3d.size)
```

```
(venv) C:\Users\ajcemca\PycharmProjects\pythonProject1>python 3D.py
3D Array:
 [[[9 7 4 7]
  [5 5 9 7]
  [7 4 8 3]]

 [[6 7 3 3]
  [4 3 1 3]
  [9 5 4 7]]]
Shape: (2, 3, 4)
Number of dimensions (ndim): 3
Total size (number of elements): 24
```